Vulnerability Assessment Report of

LIVMOR API

Sept 2021

**About L&T Technology Services:**

L&T Technology Services Limited (LTTS) is a global leader in Engineering and R&D (ER&D) services. With 399 patents filed for 51 of the Global Top 100 ER&D spenders. Our innovations speak for itself – World’s 1st Autonomous Welding Robot, Solar ‘Connectivity’ Drone, and the Smartest Campus in the World, to name a few. LTTS expertise in engineering design, product development, smart manufacturing, and digitalization touches every area of our lives. With 49 Innovation and R&D design centers globally, we specialize in disruptive technology spaces such as 5G, Artificial Intelligence, Collaborative Robots, Digital Factory, and Autonomous Transport. **LTTS is a publicly listed subsidiary of Larsen & Toubro Limited, the $18 billion Indian conglomerate operating in over 30 countries.**

**Document History:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Var. Rel.**  **No.** | **Release Date** | **Prepared By.**  **Prepared. Dt.** | **Reviewed By**  **Reviewed Dt.** | **Approved By**  **Approval Dt.** | **Remark/Revision Details** |
| 1.0 | 07-10-2021 | Niharika  Pallavi | Vishal S R | Atanu  Niyogi |  |
| 07-10-2021 | 07-10-2021 | dd-10-2021 |

**Table of Contents**

[1. Overview of the project 5](#_Toc84520980)

[2. Evidences for Unsuccessful Exploit Attempts 7](#_Toc84520981)

[2.1 Cross-site Scripting 7](#_Toc84520982)

[2.2 SQL Injection 8](#_Toc84520983)

[2.3 XPath Injection 9](#_Toc84520984)

[2.4 Invalid Types 10](#_Toc84520985)

[2.5 HTTP Method restriction 11](#_Toc84520986)

[2.6 Insecure Direct Object References (IDOR) 12](#_Toc84520987)

[2.7 Token Authorization 13](#_Toc84520988)

[3. Vulnerabilities explained in detail 14](#_Toc84520989)

[3.1 Cross-site Scripting (XSS) 14](#_Toc84520990)

[3.2 Directory Traversal 15](#_Toc84520991)

[3.3 SQL Injection Attack 17](#_Toc84520992)

[3.4 No Authentication 19](#_Toc84520993)

[3.5 Cookie without ‘Secure’ attribute 20](#_Toc84520994)

[3.6 Cookie without ‘HTTPOnly’ attribute 21](#_Toc84520995)

[3.7 CRLF Injection attack 23](#_Toc84520996)

[3.8 Insecure Direct Object References (IDOR) 24](#_Toc84520997)

[3.9 CORS Misconfiguration 25](#_Toc84520998)

[3.10 No Content Security Policy 26](#_Toc84520999)

[4. Abbreviation 28](#_Toc84521000)

[5. Appendix 28](#_Toc84521001)

# Overview of the project

L&T Technology Services (LTTS) security team has conducted Security Assessment for Livmor APIs. The purpose of the assessment is to evaluate the security posture of the Livmor APIs against common vulnerabilities. LIVMOR’s wearable platform aims to record key health parameters of the patient and showcase it to both the doctor as well as the patient. The data is processed in ways to help them monitor and maintain key parameters under prescribed levels.

**Objective of the security assessment:**

As a part of this engagement a holistic approach was taken to conduct the Vulnerability Assessment and Penetration Testing on Livmor APIs. During the engagement High, Medium, and Low severity issues were identified with respect to Livmor APIs.

**Approach**

The following approach was taken to make sure the target site is assessed against OWASP Top 10 Vulnerabilities from all possible security perspectives:

Manual Penetration Testing in tandem with penetration testing tools. Some of the tools which were used are listed below:

|  |  |
| --- | --- |
| Target site URL | <https://salesforceapi.livmor.com/userdetails>  <https://salesforceapi.livmor.com/dashboard>  <https://livmorinc--hvdev.my.salesforce.com/services/data/v45.0/sobjects/HV_Notifications__c>  https://salesforceapi.livmor.com/peripheral  <https://salesforceapi.livmor.com/reportdetails>  <https://salesforceapi.livmor.com/watchsessions>  https://salesforceapi.livmor.com/calendartask |
| Browser | Chrome, Firefox |
| Tools | BURP, Postman, ReadyAPI |

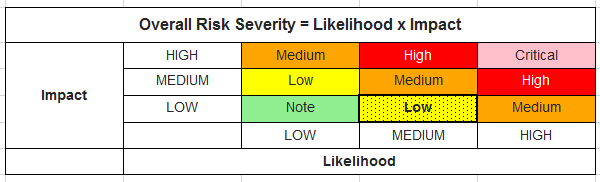
* + 2. Key Security Policies

OWASP top 10 listed vulnerabilities were used as a reference framework. The following key security aspects were checked:

* 1. Injection
  2. Broken Authentication
  3. Sensitive Data Exposure
  4. XML External Entities (XXE)
  5. Broken Access Control
  6. Security Misconfiguration
  7. Cross-Site Scripting (XSS)
  8. Insecure Deserialization
  9. Using Components with Known Vulnerabilities
  10. Insufficient Logging & Monitoring

**Summary of Findings**

The graph below shows a summary of the number of vulnerabilities found for each impact level for the API testing. Vulnerabilities found are addressed according to priority, findings, analysis, and recommendations from the assessment.



# Evidences for Unsuccessful Exploit Attempts

|  |
| --- |
| **2.1 Cross-site Scripting** |
| **Description:**  Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. |
| **Evidence:**  C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png  C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png  C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png |

|  |
| --- |
| **2.2 SQL Injection** |
| **Description:** SQL Injection Scans work through a list of predefined strings that could be used to execute arbitrary SQL code in a database, and inserts those strings into the parameters of the request. If an unexpected response is received, this is an indication that input validation has failed to remove the potentially malicious SQL strings from the parameters, and that data should be sanitized before it is used to construct SQL queries. |
| **Evidence:**  C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png  C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png  C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png |

|  |
| --- |
| **2.3 XPath Injection** |
| **Description:** XPath Injection Scans work through a list of predefined strings known to present problems for XPath parsers, and inserts those strings into the parameters of the request. If an unexpected response is received, this is an indication that input validation has failed to remove the potentially malicious strings from the parameters, and that data should be sanitized before it is used to build XPath expressions. |
| **Evidence:**  C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png  C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png  C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png |

|  |
| --- |
| **2.4 Invalid Types** |
| **Description:** An Invalid Types Scan tries to confuse the system under test by deliberately inserting incorrectly typed data into your parameters, for example, a string containing letters into a numeric field. Alerts usually indicate that you need to improve input validation and error handling. |
| **Evidence:**  C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png  C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png  C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png |

|  |
| --- |
| **2.5 HTTP Method restriction** |
| **Description:**An HTTPMethod restriction refers to use of proper HTTP method according to the operation: GET (read), POST (create), PUT/PATCH (replace/update), and DELETE (to delete a record), and respond with 403(Forbidden), 405 (Method Not Allowed), etc., if the requested method is not appropriate for the requested resource. |
| **Evidence:**  C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png  C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png |

|  |
| --- |
| **2.6 Insecure Direct Object References (IDOR)** |
| **Description:** Insecure Direct Object References occur when an application provides direct access to objects based on user supplied input. Because of this vulnerability attackers can bypass authorization and access resources directly by modifying the value of a parameter used as reference towards a system resource, for example database records of other users. This is caused by the fact that the application takes user supplied input and uses it to retrieve an object without performing sufficient authorization checks. |
| **Evidence:**  C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png |

|  |
| --- |
| **2.7 Token Authorization** |
| **Description:** Token-based authentication allows users to verify their identity and to secure sensitive data and respond with 401(Unauthorized) if the token is not valid or expired. |
| **Evidence:**  C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png |

# Vulnerabilities explained in detail

|  |  |  |  |
| --- | --- | --- | --- |
| **3.1 Cross-site Scripting (XSS)** | | | |
| **Impact** | High | **Risk Rating** | Critical |
| **Ease of Exploit** | Easy |
| **Likelihood** | High |
| **Category** | Reflected XSS into HTML context with nothing encoded | | |
| **URL/Impacted system** | <https://livmorinc--hvdev.my.salesforce.com/services/data/v45.0/sobjects/HV_Notifications__c> | | |
| **Description** | | | |
| It was observed that the API is vulnerable to cross Site Scripting (XSS). Cross-site scripting is a web security vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable application to access any of the user's data. If the victim user has privileged access within the application, then the attacker might be able to gain full control over all the application's functionality and data. | | | |
| **Impact** | | | |
| If the compromised user has elevated privileges within the application, then the impact will generally be critical, allowing the attacker to take full control of the vulnerable application and compromise all users and their data. | | | |
| **Recommendation** | | | |
| * Validate to catch potentially malicious user-provided input * Encode output to prevent potentially malicious user-provided data from triggering automatic load-and-execute behaviour by a browser * Use Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities that still occur. | | | |
| **How to recreate the Security defect** | | | |
| * Browse to – <https://livmorinc--hvdev.my.salesforce.com/services/data/v45.0/sobjects/HV_Notifications__c> * Capture traffic observe response using burp suite * Send the request to repeater and intruder and give payloads * And start the attack and observe the response | | | |
| **Evidence** | | | |
| C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png  C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **3.2 Directory Traversal** | | | |
| **Impact** | High | **Risk Rating** | High |
| **Ease of Exploit** | Easy |
| **Likelihood** | Medium |
| **Category** | Exposure of Information Through Directory Listing | | |
| **URL/Impacted system** | <https://livmorinc--hvdev.my.salesforce.com/services/data/v45.0/sobjects/HV_Notifications__c> | | |
| **Description** | | | |
| Directory traversal occurs when an attacker obtains unauthorized access to the contents of a directory or file on the server by exploiting how the server dynamically generates paths to those resources. There are traditionally two ways to refer to resources on a file system:  \* The absolute path to the resource  \* An abstract path which contains control characters that alter the path through directory change (including, but not limited to, ../,..\, /,\, and ~/)  The application currently uses untrusted data to construct the path and filename of a given resource, which is then returned to the user. Since the application does not sanitize or otherwise validate this input, the user is able to supply an abstract path which may refer to an unauthorized resource on the server. | | | |
| **Impact** | | | |
| Directory traversal results in an attacker gaining unauthorized access to the file system by manipulating input used by the application to construct a pathname. | | | |
| **Recommendation** | | | |
| * Instead of using a user-supplied filename to access the file, the application should maintain a mapping from integer keys to file names, ensuring that the supplied value is both an integer and corresponds to an actual file. In the case that a value does not have a corresponding file, the application should return a generic error message. This will provide whitelist validation for the files that exist in the application. | | | |
| **How to recreate the Security defect** | | | |
| * Browse to: <https://livmorinc--hvdev.my.salesforce.com/services/data/v45.0/sobjects/HV_Notifications__c> * Capture the request with burp suite. * Send the request to repeater and change the URL while adding /id in the end of URL * After that you can see the information/ output. | | | |
| **Evidence** | | | |
| **Graphical user interface, text, application, email  Description automatically generated** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **3.3 SQL Injection Attack** | | | |
| **Impact** | High | **Risk Rating** | High |
| **Ease of Exploit** | Easy |
| **Likelihood** | Medium |
| **Category** | Exposure of Sensitive data/ Information | | |
| **URL/Impacted system** | <https://livmorinc--hvdev.my.salesforce.com/services/data/v45.0/sobjects/HV_Notifications__c> | | |
| **Description** | | | |
| SQL injection is one of the most common attack mechanisms utilized by attackers to steal sensitive data from organizations.  While SQL Injection can affect any data-driven application that uses a SQL database, **it is most often used to attack web sites.** SQL Injection is a code injection technique that hackers can use to insert malicious SQL statements into input fields for execution by the underlying SQL database.  This technique is made possible because of improper coding of vulnerable web applications. | | | |
| **Impact** | | | |
| A successful SQL injection attack can result in unauthorized access to sensitive data, such as  passwords, credit card details, or personal user information. | | | |
| **Recommendation** | | | |
| * The most basic SQL injections can be prevented by **filtering input**. However, it will not stop more complex attacks. * To prevent maximum SQL injection damage, it is recommended to separate your sensitive data and store it in **multiple databases.** * Customize your error messages so they don't give away any information about the **structure** of your database to a potential intruder. | | | |
| **How to recreate the Security defect** | | | |
| * Browse to: <https://livmorinc--hvdev.my.salesforce.com/services/data/v45.0/sobjects/HV_Notifications__c> * Capture the request with burp suite. * Forward the request to intruder. * Give (1000-4000) payloads of SQL injection in the Credential position * After successful attack, we can get other details. | | | |
| **Evidence** | | | |
| C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png  C:\Users\40012873\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled.png | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **3.4 No Authentication** | | | |
| **Impact** | High | **Risk Rating** | High |
| **Ease of Exploit** | Easy |
| **Likelihood** | Medium |
| **Category** | Information Disclosed | | |
| **URL/Impacted system** | https://salesforceapi.livmor.com/reportdetails  https://salesforceapi.livmor.com/dashboard | | |
| **Description** | | | |
| Token-based authentication allows users to verify their identity and to secure sensitive data. While performing manual penetration testing on Livmor APIs, it was observed that the APIs calls are not authorized. | | | |
| **Impact** | | | |
| The attack can result in unauthorized access to sensitive data/ information, such as credential username, password etc. | | | |
| **Recommendation** | | | |
| **It is recommended to use standard authentication for all API calls.** | | | |
| **How to recreate the Security defect** | | | |
| * Browse to the API: <https://salesforceapi.livmor.com/reportdetails>   OR <https://salesforceapi.livmor.com/dashboard>   * Observe that in response you receive the expected output without any authentication. | | | |
| **Evidence** | | | |
| A screenshot of a computer  Description automatically generated  A screenshot of a computer  Description automatically generated | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **3.5 Cookie without ‘Secure’ attribute** | | | |
| **Impact** | High | **Risk Rating** | High |
| **Ease of Exploit** | Difficult |
| **Likelihood** | Medium |
| **Category** | Sensitive Cookie in HTTPS Session Without 'Secure' Attribute | | |
| **URL/Impacted system** | <https://livmorinc--hvdev.my.salesforce.com/services/data/v45.0/sobjects/HV_Notifications__c> | | |
| **Description** | | | |
| While performing manual penetration testing, it was observed that the Secure attribute for sensitive cookies in HTTPS sessions is not set, which could cause the user agent to send those cookies in plaintext over an HTTP session. If secure attribute is not set for cookie, the cookie will go on HTTP communication which will enable the attacker to do man in middle attack and sniff session cookie value. | | | |
| **Impact** | | | |
| If secure attribute is not set for cookie, the cookie will go on HTTP communication which will enable the attacker to do man in middle attack and sniff session cookie value. | | | |
| **Recommendation** | | | |
| * The 'Secure' attribute should be set on all cookies that contain sensitive values such as session IDs. Many application servers allow this to be done declaratively or through an administration interface; other application servers require the attribute to be set programmatically. The mitigation details depend on the application server being used. | | | |
| **How to recreate the Security defect** | | | |
| * Browse to: <https://livmorinc--hvdev.my.salesforce.com/services/data/v45.0/sobjects/HV_Notifications__c> * Capture traffic observe response * Found the response without set cookie flag | | | |
| **Evidence** | | | |
| **A screenshot of a computer  Description automatically generated** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **3.6 Cookie without ‘HTTPOnly’ attribute** | | | |
| **Impact** | High | **Risk Rating** | High |
| **Ease of Exploit** | Difficult |
| **Likelihood** | Medium |
| **Category** | Sensitive Cookie Without 'HttpOnly' Flag | | |
| **URL/Impacted system** | <https://livmorinc--hvdev.my.salesforce.com/services/data/v45.0/sobjects/HV_Notifications__c> | | |
| **Description** | | | |
| While performing manual penetration testing, it was observed that the HttpOnly flag is not set on cookies containing sensitive information. The HttpOnly flag is a flag set as part of a Set-Cookie header to prevent that cookie’s value from being read or set by client-side in modern browsers. Without the HttpOnly attribute, the cookie's value can be accessed by client-side scripts like JavaScript. | | | |
| **Impact** | | | |
| An attacker who can intercept a victim's session identifier would gain complete control of that user's session, allowing the attacker to access any information and perform any action with the victim’s privileges. | | | |
| **Recommendation** | | | |
| Set the 'HttpOnly' attribute when sending Set-Cookie headers in responses. This can be done either programmatically or through deployment descriptors provided by various frameworks. Note that in many newer technologies such as newer versions of Apache Tomcat web server, the HTTP Only attribute for sensitive cookies such as session cookies, is set to true by default. | | | |
| **How to recreate the Security defect** | | | |
| * Browse to: <https://livmorinc--hvdev.my.salesforce.com/services/data/v45.0/sobjects/HV_Notifications__c> * Send the request and observe the response * Found the response without HTTP Only flag | | | |
| **Evidence** | | | |
| **A screenshot of a computer  Description automatically generated** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **3.7 CRLF Injection attack** | | | |
| **Impact** | High | **Risk Rating** | High |
| **Ease of Exploit** | Easy |
| **Likelihood** | Medium |
| **Category** | Exposure of Information Through Header change | | |
| **URL/Impacted system** | <https://salesforceapi.livmor.com/dashboard> | | |
| **Description** | | | |
| CRLF refers to the special character elements "Carriage Return" and "Line Feed." These elements are embedded in HTTP headers and other software code to signify an End of Line (EOL) marker. Exploits occur when an attacker can inject a CRLF sequence into an HTTP stream. By introducing this unexpected CRLF injection, the attacker can maliciously exploit CRLF vulnerabilities to manipulate the web application's functions. A more formal name for CRLF injection is Improper Neutralization of CRLF Sequences. Because CRLF injection is frequently used to split HTTP responses, it can also be designated as HTTP Response Splitting or Improper Neutralization of CRLF Sequences in HTTP Headers. | | | |
| **Impact** | | | |
| CRLF results in an attacker gaining unauthorized access to modify application data, compromising integrity and enabling the exploitation of the vulnerabilities such as: XSS, Proxy and web server cache poisoning, website defacement, Hijacking the session etc. | | | |
| **Recommendation** | | | |
| * Always follow the rule of never trusting user input. * Sanitize and neutralize all user-supplied data or properly encode output in HTTP headers that would otherwise be visible to users to prevent the injection of CRLF sequences and their consequences. | | | |
| **How to recreate the Security defect** | | | |
| * Browse to: <https://salesforceapi.livmor.com/dashboard> * Capture the request with burp suite. * Send the request to repeater and change the Content-Type into text/html\r\n * After that observe the response | | | |
| **Evidence** | | | |
| **Graphical user interface, text, application  Description automatically generated** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **3.8 Insecure Direct Object References (IDOR)** | | | |
| **Impact** | High | **Risk Rating** | High |
| **Ease of Exploit** | Easy |
| **Likelihood** | Medium |
| **Category** | Improper Privilege Management | | |
| **URL/Impacted system** | <https://livmorinc--hvdev.my.salesforce.com/services/data/v45.0/sobjects/HV_Notifications__c> | | |
| **Description** | | | |
| Insecure direct object references (IDOR) is a cybersecurity issue that occurs when an API developer uses an identifier for direct access to an internal implementation object but provides no additional access control and/or authorization checks. For example, an IDOR vulnerability would happen if the URL of a transaction could be changed through client-side user input to show unauthorized data of another transaction. | | | |
| **Impact** | | | |
| Exposure of Confidential Information: When the attacker will have control over your account via this vulnerability, it is obvious that an attacker will be able to come across your personal information. | | | |
| **Recommendation** | | | |
| • Developers should avoid displaying private object references such as keys or filenames.  • Validation of Parameters should be properly implemented.  • Verification of all the Referenced objects should be done.  • Tokens should be generated in such a way that it should only be mapped to the user and should not be public. | | | |
| **How to recreate the Security defect** | | | |
| * Browse to the API: <https://livmorinc--hvdev.my.salesforce.com/services/data/v45.0/sobjects/HV_Notifications__c> * Change the Patient Id position parameter and send the request * After successful attack, we get the desirable output | | | |
| **Evidence** | | | |
| **Graphical user interface, text, application  Description automatically generated** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **3.9 CORS Misconfiguration** | | | |
| **Impact** | Medium | **Risk Rating** | Medium |
| **Ease of Exploit** | Moderate |
| **Likelihood** | Medium |
| **Category** | Permissive Cross-domain Policy with Untrusted Domains | | |
| **URL/Impacted system** | <https://salesforceapi.livmor.com/dashboard> | | |
| **Description** | | | |
| Access-allow control attribute is incorrectly set using wildcards such as (\*) under which domains can request resources. It enables controlled access to resources which are outside of the given domain. It adds flexibility to Same Origin Policy (SOP) . | | | |
| **Impact** | | | |
| * This is usually set as default, which means any domain can access resources on this site.   Able to steal confidential and sensitive information. | | | |
| **Recommendation** | | | |
| * Set the Access-allow control header to validated and whitelisted websites * To implement CORS securely, you need to associate a validation list with Access-Control-Allow-Origin that identifies which specific domains can access resources. Then your application can validate against this list when a domain requests access. | | | |
| **How to recreate the Security defect** | | | |
| * Browse to - <https://salesforceapi.livmor.com/dashboard> * Capture traffic observe response using burp suite * Send the request to repeater and observe the response. | | | |
| **Evidence** | | | |
| **Graphical user interface, text, application, email  Description automatically generated** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **3.10 No Content Security Policy** | | | |
| **Impact** | Low | **Risk Rating** | Low |
| **Ease of Exploit** | Difficult |
| **Likelihood** | Medium |
| **Category** | Improper Restriction of Rendered UI Layers or Frames | | |
| **URL/Impacted system** | <https://salesforceapi.livmor.com> | | |
| **Description** | | | |
| During the HTTP traffic analysis of the Livmor APIs, it was observed that the server does not support Content Security Policy. Content Security Policy is a standard that helps protect against various content injection attacks like cross site scripting. While the victim is interacting with seemingly harmless web pages. | | | |
| **Impact** | | | |
| Without a Content Security Policy, an attacker can perform content injection attacks if data from the service is displayed in a browser | | | |
| **Recommendation** | | | |
| Enabling Content Security Policy response header to all HTTP server responses helps in preventing content injection attacks. While adding Content Security Policy it must be set correctly specifying the locations from which content can be loaded. Content-Security-Policy: <Policy-directive>; | | | |
| **How to recreate the Security defect** | | | |
| * Browse to - <https://csp-evaluator.withgoogle.com/> * Enter the URL – <https://salesforceapi.livmor.com> * Click on check CSP | | | |
| **Evidence** | | | |
| **Graphical user interface, text, application  Description automatically generated** | | | |

# Abbreviation

|  |  |
| --- | --- |
| APP | Application |
| HTML | Hyper Text Mark-up Language |
| HTTP | Hypertext transfer protocol |
| LTTS | Larsen & Toubro Technology Services |
| VAPT | Vulnerability Assessment and Penetration Testing |
| SOP | Same Origin Policy |
| OWASP | Open Web Application Security Project |
| IDOR | Insecure direct object references |
| SOP | Same Origin Policy |
| CRLF | Carriage Return Line Feed |
| CSP | Content Security Policy |
| CORS | Cross-origin resource sharing |
| XSS | Cross-site Scripting |
| URL | Uniform Resource Locator |
| XSS | Cross Site Scripting |
| XXE | XML External Entities |

# 

# Appendix

Vulnerability scan reports:





